УДК 004.21

Студ. Е.С. Сафонов
Науч. рук. преп.-стажер А.С. Север
(Кафедра программной инженерии, БГТУ)

# СБОРЩИК МУСОРА C++

В данной научной работе рассматриваются понятие сборщика мусора. Были разобраны два основных вида сборщика. Был описан сборщик с маркировкой (MarkSweepCollector), а также CopyCollector. MarkSweepCollector при создании объекта записывает ссылку на объект в собственную таблицу ссылок и после того как ссылка перестаёт существовать, то память занимаемая объектом удаляется. CopyCollector имеет похожий принцип только при сборке он перемещает все нужные данные из фрагмента памяти в другой свободный фрагмент. После перемещения данных старый фрагмент полностью очищается. Был описаны достоинства и недостатки для каждого алгоритма сборки мусора. Далее был описан алгоритм для вида CopyCollector. Так же алгоритм был написан на языке C++, потому что для сборщика важна высока эффективность, чтобы за максимально короткий срок выполнить очистку, а также с помощью C++ разработчик имеет полный контроль над памятью, что позволяет предотвратить утечки памяти. А также C++ имеет мощные инструменты для реализации сборки.

В работе описан полный алгоритм, который был разделён на функции, таких как функция инициализации кучи, выделения памяти под объект, перемещения по блокам памяти и функция самой сборки. В качестве объектов был выбрано дерево поиска и добавлены функции работы с ним (удаление, поиск, добавление, вывод). В главной функции main продемонстрированно создания дерева, удаления элементов и вывод результата работы сборщика, путём вывода адресов каждого элемента дерева. Полный исходный код алгоритма размещён на удаленном репозитории по ссылке «https://github.com/awssed/GarbageCollector». Тема сборщика мусора является актуальной в свете постоянно растущих требований к производительности и оптимизации работы приложений на данном языке программирования. Разработка сборщика мусора может быть полезна для различных проектов, в которых требуется оптимизировать использование памяти и избежать утечек. Более того, изучение алгоритмов сборки мусора может помочь разработчикам лучше понимать внутреннее устройство языка и использовать его более эффективно.

В свете того, что в настоящее время многие разработчики сталкиваются с проблемами управления памятью и необходимостью оптимизации работы приложений, тема сборщика мусора на C++ является актуальной и имеет потенциальную ценность для широкого круга читателей.